CSCE 750
Final Exam Answer Key
Wednesday December 7, 2005

Do all problems. Put your answers on blank paper or in a test booklet. There are 100 points total in the exam. You have 180 minutes.

Please note the following:

- This exam is open book, open notes, but no electronic devices.
- There are eleven regular questions, plus one question that counts for Quiz 13. All regular questions are worth the same amount, regardless of their difficulty. Your lowest score on a regular question will be dropped. This makes each regular question worth 4% of your grade. The quiz question is worth 5% of your grade.
- In general, unless I say otherwise, you need not show your scratch work to get full credit for a correct answer. If your answer is incorrect, however, showing your work may earn you partial credit.
- You may take as given any fact proved in either of the two textbooks (CLRS or GJ).

1. Show that the sum
$$\sum_{n=2}^{\infty} \frac{1}{n \lg n}$$
diverges. (Cf. CLRS A.2-1)

Answer There are at least two "standard" ways of doing this that I know of:

(a) Split the sum up into blocks of size 1, 2, 4, 8, 16, etc. and lower-bound each term with the smallest term in its block:

$$\begin{aligned}
\sum_{n=2}^{\infty} \frac{1}{n \lg n} &= \frac{1}{2\lg 2} + \left(\frac{1}{3\lg 3} + \frac{1}{4\lg 4}\right) + \left(\frac{1}{5\lg 5} + \cdots + \frac{1}{8\lg 8}\right) + \cdots \\
&\geq \frac{1}{2\lg 2} + \left(\frac{1}{4\lg 4} + \frac{1}{4\lg 4}\right) + \left(\frac{1}{8\lg 8} + \cdots + \frac{1}{8\lg 8}\right) + \cdots \\
&= \frac{1}{\lg 2}\left(\frac{1}{2}\right) + \frac{1}{\lg 4}\left(\frac{1}{4} + \frac{1}{4}\right) + \frac{1}{\lg 8}\left(\frac{1}{8} + \cdots + \frac{1}{8}\right) + \cdots \\
&= \frac{1}{2}\left(\frac{1}{\lg 2} + \frac{1}{\lg 4} + \frac{1}{\lg 8} + \cdots\right) \\
&= \frac{1}{2}\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots\right) \\
&= \frac{1}{2}\sum_{k=1}^{\infty} \frac{1}{k} \\
&= \infty.
\end{aligned}$$

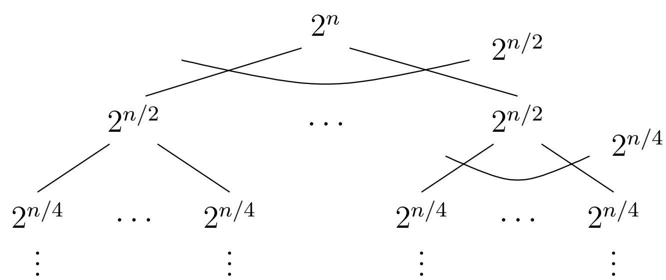
(b) Use an integral approximation with a change of variables $u = \ln x$. Let $k \geq 2$ be an integer. Noting that $1/(n \ln n)$ is monotone decreasing as a function of n, we have

$$\begin{aligned}
\sum_{n=2}^{k} \frac{1}{n \lg n} &= (\ln 2) \sum_{n=2}^{k} \frac{1}{n \ln n} \\
&\geq (\ln 2) \int_2^{k+1} \frac{dx}{x \ln x} \\
&= (\ln 2) \int_{\ln 2}^{\ln(k+1)} \frac{du}{u} \\
&= (\ln 2)[\ln\ln(k+1) - \ln\ln 2].
\end{aligned}$$

Taking the limit as $k \to \infty$ shows that the sum diverges, albeit very slowly.

2. Draw a recursion tree and give a tight asymptotic bound (i.e., using $\Theta(\cdot)$ notation) on the solution of the recurrence
$$T(n) = 2^{n/2}T(n/2) + 2^n.$$
You need not prove your answer. (Cf. CLRS 4.2-3)

Answer Here is the tree.



The tree is full and balanced with depth $\lg n$. For each nonleaf node x in the tree, the value at x is equal to the sum of the values at all of x's children. (This can be seen easily by induction.) Thus, the sum of values at each level is the same, namely 2^n. Therefore, $T(n) = \Theta(2^n \lg n)$.

3. During the running of the procedure RANDOMIZED-QUICKSORT, how many calls are made to the random-number generator RANDOM in the worst case? Give your answer using Θ-notation. (Cf. CLRS 7.3-2)

   Answer Note that "worst case" here refers to a case that maximizes the number of calls to RANDOM. This may or may not coincide with a case that maximizes the total running time.

   There are $\Theta(n)$ calls to RANDOM in the worst case. This is also true for the best case. Each call to random chooses a pivot. Each pivot value is chosen only once, after which it does not participate in further partitions. With n items in the list, there can be at most n pivots, hence at most n calls to RANDOM. There is clearly a case where there are $n-1$ calls to PARTITION (and hence $n-1$ calls to RANDOM), namely, when the pivot is always chosen to be an extreme value in the list to be partitioned.

4. Explain why any comparison-based algorithm that finds the median of a list of n numbers must always make at least $n-1$ comparisons, even in the best case. (Cf. CLRS 8.1-1)

   Answer Note: we will assume that duplicate values are not allowed in the input list. The statement is false otherwise. (For example, on input $\langle a_1, a_2, a_3, a_4, a_5 \rangle$, an algorithm might compare a_1 with a_2, a_2 with a_3, and a_3 with a_1 and thus find that $a_1 = a_2 = a_3$; the median must be this common value, and so the algorithm outputs a_1 with only three comparisons.)

   The idea is that with fewer than $n-1$ comparisons, there are groups of elements with no comparisons made between them. By tweaking one or another of these groups, we can change the median without changing the output of A, thus making A incorrect on at least one input.

Here is a more careful proof using this idea.

Let A be any comparison-based algorithm that outputs one of the values in its input list (this is certainly true of any comparison-based median-finding algorithm). Suppose that there is some n-element input list $\langle a_1, \ldots, a_n \rangle$ on which A makes strictly fewer than $n-1$ comparisons. We'll explain why there must be some input list $\langle b_1, \ldots, b_n \rangle$ on which A does not output the median (and hence A is not a median-finding algorithm).

Consider the decision tree of A for inputs of size n, and let p be the path taken by A on input $\langle a_1, \ldots, a_n \rangle$. By assumption, there are at most $n-2$ comparisons made along p. I claim that the elements of the input list can be divided into two disjoint, nonempty sets X and Y such that no element of X is compared with any element of Y along the path p. [Consider the input elements as vertices of an undirected graph. For each comparison of elements u and v along p, draw an edge connecting u and v. The resulting graph has at most $n-2$ edges, and so it is not connected. Let X be the vertices in any connected component of the graph, and let Y be the rest of the vertices.] Let M be the largest value in the input list $\langle a_1, \ldots, a_n \rangle$, and let m be the smallest value. Set $R = M - m$, the range of values in the list $\langle a_1, \ldots, a_n \rangle$. With this list as input, algorithm A outputs some element a_i at the end of path p. Suppose that $a_i \in X$. (The argument for $a_i \in Y$ is the same but with the roles of X and Y reversed.) Let $\langle b_1, \ldots, b_n \rangle$ be the list that results from $\langle a_1, \ldots, a_n \rangle$ by adding R to every element in Y, and let $\langle c_1, \ldots, c_n \rangle$ be similar except that we subtract R from every element in Y (the elements in X stay the same). Note the following two facts:

- $\langle b_1, \ldots, b_n \rangle$ and $\langle c_1, \ldots, c_n \rangle$ have different medians. This follows from the fact that all Y-elements of the former are larger than all X-elements, and vice versa for the latter list.
- A outputs the same value (a_i) on the three inputs $\langle a_1, \ldots, a_n \rangle$, $\langle b_1, \ldots, b_n \rangle$, and $\langle c_1, \ldots, c_n \rangle$. This is because all comparison along the path p give the same results among the three inputs, so A takes path p for all three, and outputs the same element a_i, which, being in X, is the same in all three inputs.

Thus A does not output the median for at least one of the input lists $\langle b_1, \ldots, b_n \rangle$ and $\langle c_1, \ldots, c_n \rangle$.

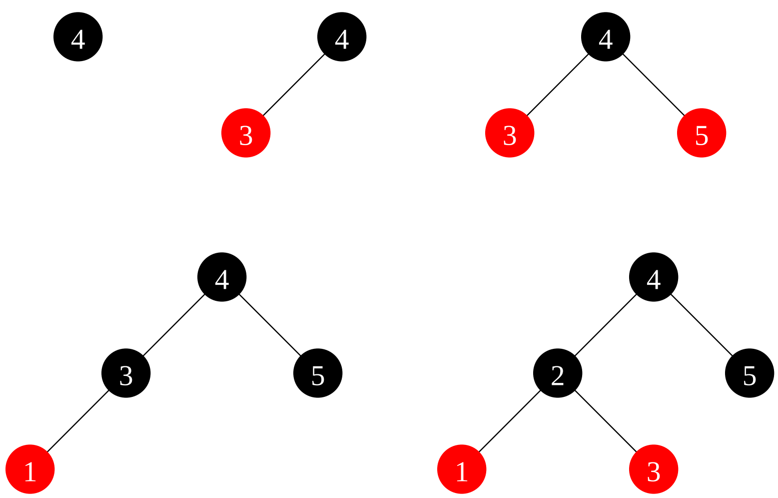
5. An ordinary six-sided die is rolled three times in a row. What is the probability that the values come in strictly increasing order? (Cf. CLRS C.2-3)

Answer Here's the slick way. There are 6^3 possible outcomes (i.e., elementary events) in the sample space $\Omega = D \times D \times D$, where $D = \{1, 2, 3, 4, 5, 6\}$ is the sample space for a single die roll. The set of outcomes where the three numbers occur in strictly increasing order can be placed in one-to-one correspondence with the set of three-element subsets of D. Thus there are $\binom{6}{3}$ many possibilities. Each possible way is equally likely, with probability $1/6^3 = 1/216$. Thus the probability is

$$\frac{\binom{6}{3}}{6^3} = \frac{5}{54}.$$

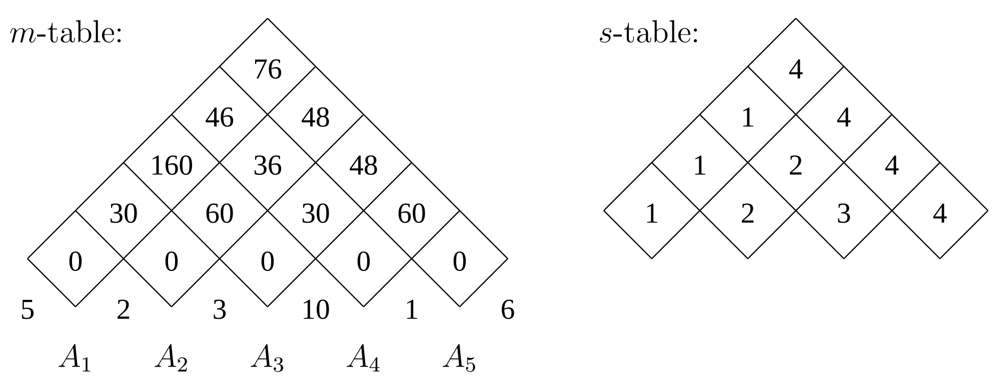
6. Show the red-black trees that result after successively inserting the keys $4, 3, 5, 1, 2$ in that order into an initially empty red-black tree. Be sure to show the color of each node. (Cf. CLRS 13.3-3)

Answer There are five trees in all.



7. Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 5, 2, 3, 10, 1, 6 \rangle$. (Cf. CLRS 15.2-1)

Answer The optimal parenthesization is $(A_1(A_2(A_3A_4)))A_5$. Here are the tables.



8. A sequence of n operations is performed on a data structure. The ith operation costs $\sqrt{i}$ if i is a perfect square, and 1 otherwise. Using any method you like (e.g., aggregate analysis, accounting method, or potential method), determine the amortized cost per operation. Show your work. (Cf. CLRS 17.1-3, 17.3-2)

Answer

Aggregate Method Let $n > 0$ be an integer. The total cost of the first n operations is

$$C = n + \sum_{i=1}^{\lfloor\sqrt{n}\rfloor}(i-1) \le n + \frac{\sqrt{n}(\sqrt{n}-1)}{2} < 3n/2.$$

The amortized cost per operation is C/n, which is thus at most $3/2 = O(1)$.

Potential Method The idea is that the distance between two successive perfect squares, say i^2 and $(i+1)^2$, is $2i+1$, so we can amortize the cost of the next expensive operation by tacking on a constant additional charge (at least 1/2 per operation) to the cheap operations. Let c be any constant greater than or equal to 1/2, and for $i = 1, 2, 3, \ldots$ define $\Phi(i) = c(i - \lfloor\sqrt{i}\rfloor^2)$. That is, if we let m be the largest integer such that $m^2 \le i$ and let k be such that $i = m^2 + k$, then $\Phi(i) = ck$. We have two cases.

$k > 0$. The amortized cost of the ith operation is

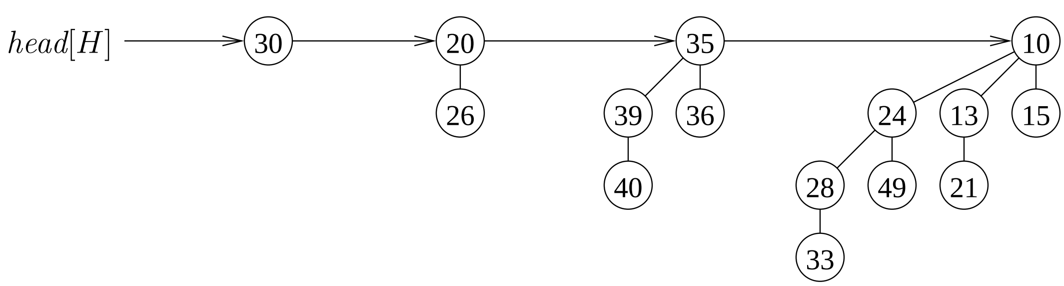
$$1 + \Phi(i) - \phi(i-1) = 1 + ck - c(k-1) = 1 + c = O(1).$$

$k = 0$. We have $i = m^2$, and so the amortized cost of the ith operation is

$$m + \Phi(i) - \Phi(i-1) = m + 0 - c[m^2 - 1 - (m-1)^2] = (1-2c)m + 2c = O(1).$$

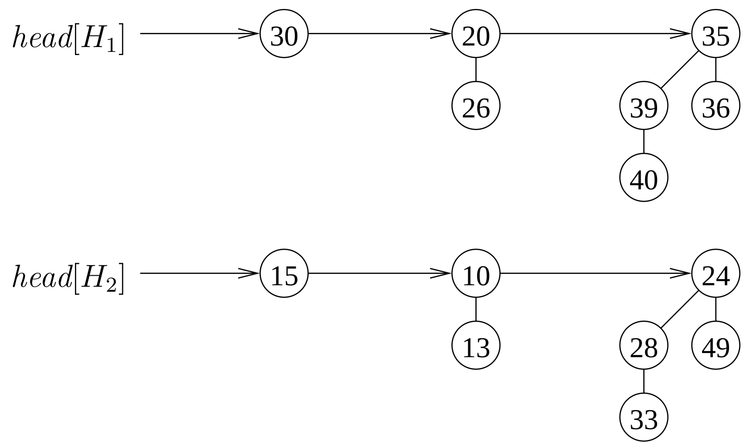
Thus the amortized cost per operation is $O(1)$.

9. Show the binomial min-heap that results when the element 21 is removed of H (below):

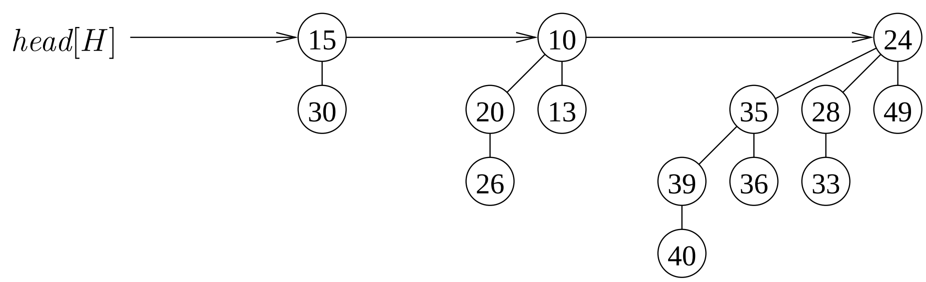


(Cf. CLRS 19.2-3)

Answer We must merge the two heaps



to get



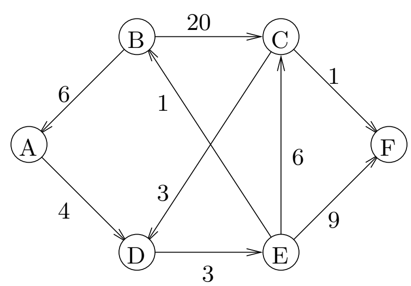
10. An undirected graph $G = (V, E)$ is *bipartite* if V can be partitioned into two disjoint sets V_1 and V_2 with $V = V_1 \cup V_2$ such that every edge in E connects a vertex from V_1 with a vertex from V_2. [This may not be relevant, but it is well-known that G is bipartite iff G has no cycles of odd length.] Explain in words how to use Breadth-First Search to find such a V_1 and V_2 (which may not be unique) if G is bipartite, or else output, "G is not bipartite." Keep your description high-level English, without pseudocode. The procedure you describe need not be the most efficient, but it should at least run in polynomial time. [Hint: Suppose G is bipartite with vertex partition V_1, V_2. Fix any vertex $v \in V$. Suppose WLOG that $v \in V_1$. What can you say about the (unweighted) distance between v and any vertex in V_1? Between v and any vertex in V_2?] (Cf. 22.2-6)

Answer The idea is that if G is bipartite with partition V_1, V_2 and $v \in V_1$, then the unweighted distance between v and any vertex in V_1 is even (or possibly ∞), and the distance between v and any vertex in V_2 is odd (or possibly ∞).

Choose any $v \in V$. Run BFS to find the unweighted distance from v to other vertices. If not all vertices are reachable from v, then repeat BFS with an unreached vertex, and so on, until all vertices have finite distances. Let V_1 be the set of vertices whose distances are even, and let V_2 be the set of vertices whose distances are odd. Now, for each pair of distinct vertices $u, w \in V_1$, check if $(u, w) \in E$, and do the same for each pair of distinct vertices in V_2. If any adjacencies are found this way, say that G is not bipartite. Otherwise, output V_1 and V_2.

Here's an alternate approach that works. Starting with V_1 and V_2 being empty, do the following repeatedly until all vertices are placed into either V_1 or V_2: Starting with an unplaced vertex v, put v into V_1 and then run BFS from v. During the BFS, whenever a vertex w is updated as a result of being adjacent to u, check if u and w are in the same partition (either both in V_1 or both in V_2). If yes, stop immediately and output, "G is not bipartite." Otherwise, place w into the opposite set (the one that does not contain u) if it is not already in there. Finally, return V_1 and V_2.

11. Give the final d and π values of the vertices obtained by running Dijkstra's algorithm on the directed graph below with source A:



(Cf. CLRS Exercise 24.3-1)

Answer

v	$d[v]$	$\pi[v]$
A	0	nil
B	8	E
C	13	E
D	4	A
E	7	D
F	14	C

Quiz 13 credit: Recall the problem HAMILTONIAN CIRCUIT (HC) that we showed in class to be NP-complete:

Instance: An undirected graph G.
Question: Is there a cycle in G that includes each vertex of G exactly once?

Consider the following problem:

HAMILTONIAN PATH (HP)
Instance: An undirected graph G.
Question: Is there path in G that includes each vertex of G exactly once?

HP is clearly in NP. Show that HP is NP-complete by giving a polynomial reduction from HC to HP. That is, show how to easily transform an arbitrary undirected graph G into an undirected graph G' such that G has a Hamiltonian circuit if and only if G' has a Hamiltonian path. You may assume that G has at least three vertices. [Hint: Get G' by adding a constant number of vertices to G along with some new edges, and possibly remove some edges from G.]

Answer: Given an arbitrary undirected graph G with at least three vertices, we construct G' as follows: Choose an arbitrary vertex v of G (it does not matter which one). Let N (the neighborhood of v) be the set of all vertices adjacent to v. Add three new vertices a, b, c to the vertex set of G. Add edges between a and v, between b and c, and between c and each vertex in N. Let G' be the resulting graph. This transformation is clearly polynomial-time.

Suppose G has a Hamiltonian circuit p. This circuit must pass through v and two of v's neighbors, $x, y \in N$. By removing (v, x) from p and adding the edges (a, v), (b, c), and (c, x) to p, we clearly get a Hamiltonian path in G'. Conversely, suppose p is a Hamiltonian path in G'. The path p must be of the form $\langle a, v, \ldots, x, c, b\rangle$ for some $x \in N$, because a and b both have degree one. Then the subpath $p' = \langle v, \ldots, x\rangle$ of p must be a Hamiltonian path in G. Also, v and x are adjacent in G, but the edge (x, v) is not in p' since G has at least three vertices. So adding the edge (x, v) to the end of p' makes it a Hamiltonian circuit in G.

We have shown that G has a Hamiltonian circuit if and only if G' has a Hamiltonian path. Thus the map $G \mapsto G'$ is a polynomial reduction from HC to HP.